Project 2

Clustering Algorithms

Sagar Dhamija (sagardha : 50169364)

Varun Khandelwal (varunkha : 50168936)

Contents

Clustering

A very common task in the field of data mining is data analysis which can be implemented by grouping a set of objects into subsets such that all similar elements are clustered within a single group. From the application point of view this process of clustering similar objects together can be beneficial in many ways:

- Identifying the centers of the cells given a medical image of a group of cells.
- Locating a user's frequently visited places by analyzing their GPS data of mobile phones and then applying clustering algorithm to detect the places within a certain radius
- Structuring any arbitrary unlabeled data by identifying any similarity or pattern within the data.

K Means

k-means clustering, that stems from signal processing is a method of vector quantization, and is extremely popular in the field of data mining for cluster analysis. K-means stores k centroids that it uses to define clusters. A point is considered to be in a particular cluster if it is closer to that cluster's centroid than any other centroid.

K-Means finds the best centroids by following two iterative repetitive steps

(1) assigning data points to clusters based on the current centroids

(2) choosing centroids (points which are the center of a cluster) based on the current assignment of data points to clusters.

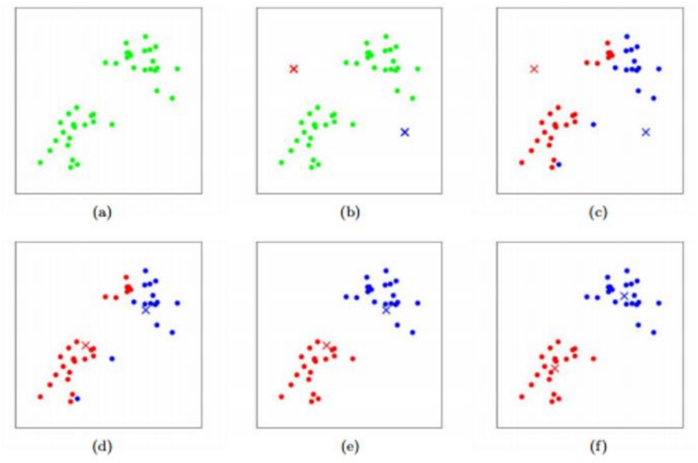


Figure 1: K-means algorithm. Training examples are shown as dots, and cluster centroids are shown as crosses. (a) Original dataset. (b) Random initial cluster centroids. (c-f) Illustration of running two iterations of k-means. In each iteration, we assign each training example to the closest cluster centroid (shown by "painting" the training examples the same color as the cluster centroid to which is assigned); then we move each cluster centroid to the mean of the points assigned to it. Images courtesy of Michael Jordan.

Mathematical Background:

The k-means algorithm takes as input a dataset X of N points, together with an additional parameter K specifying how many clusters to create. The output is a set of K cluster centroids and a labeling of X that assigns each of the points in X to a unique cluster. All points within a cluster are closer in distance to their centroid than they are to any other centroid.

The mathematical condition for the K clusters   and the K centroids   can be expressed as:

Minimize $\sum_{k=1}^{K} \sum_{x_n \in C_k} ||x_n - \mu_k||^2$ with respect to C_k, μ_k.

Lloyd's algorithm

Finding a solution to a clustering problem like this is NP hard. However, an iterative method popularly known as Lloyd's algorithm can be used that converges in a few iterations to provide efficient clustering.

The procedure oscillates between two operations:

(1) Once a set of centroids   is available, the clusters are updated to contain the points closest in distance to each centroid.

(2) Given a set of clusters, the centroids are recalculated as the means of all points belonging to a cluster.

$$C_k = \{x_n : ||x_n - \mu_k|| \leq \text{ all } ||x_n - \mu_l||\} \qquad (1)$$

$$\mu_k = \frac{1}{C_k} \sum_{x_n \in C_k} x_n \qquad (2)$$

Advantages:

- For huge variables and small K values, k-means clustering runs computationally faster.
- K-means clustering produces tighter clusters if the clusters are globular.

Disadvantages:

- The major difficulty arises in choosing the appropriate centroids.
- At times it may happen that the algorithm may not converge if inappropriate clusters are chosen.

- Different initial partitions can result in different final clusters.

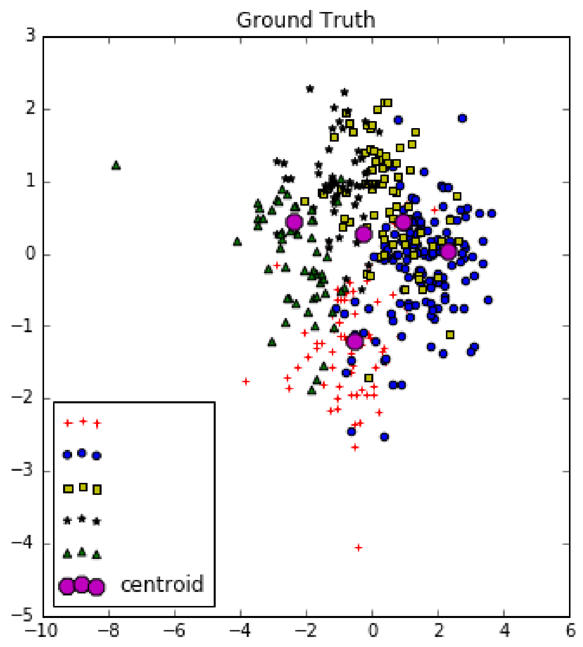
Our Implementation:

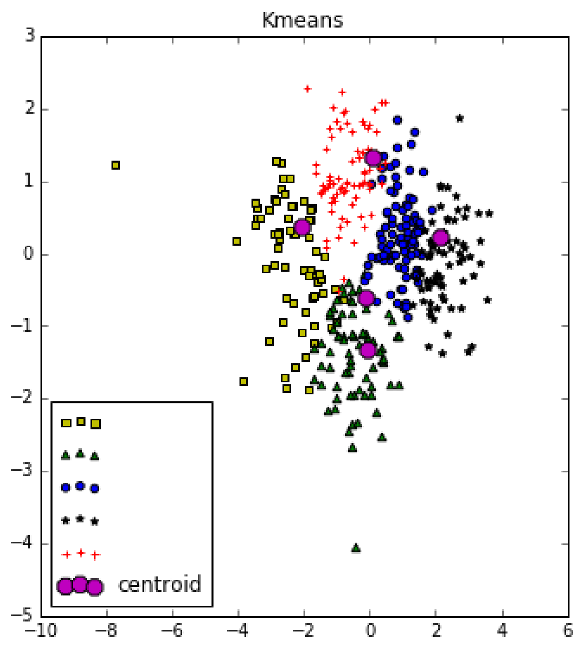
1. Coding Language: Python
2. Read the file and split the data into gene data, labels and data_id.
3. Accept number of clusters from the user
4. Function calScore that calculates RAND and JACCARD scores.
5. The main kmeans logic:
    a. Random data rows were selected as initial centroids
    b. Distance of every point was calculated against every cluster
    c. Every point was assigned to the cluster which was the nearest to it.
    d. Every clusters attribute level mean was computed and this mean was assigned as the new centroid.
    e. Repeat steps a-d till convergence or till 1000 iterations.
6. We have performed step 5 multiple times and retained the best result to minimize the chance of a bad result due to random chance.
7. Finally, we performed dimensionality reduction using PCA and displayed the results.

Result Visualization:

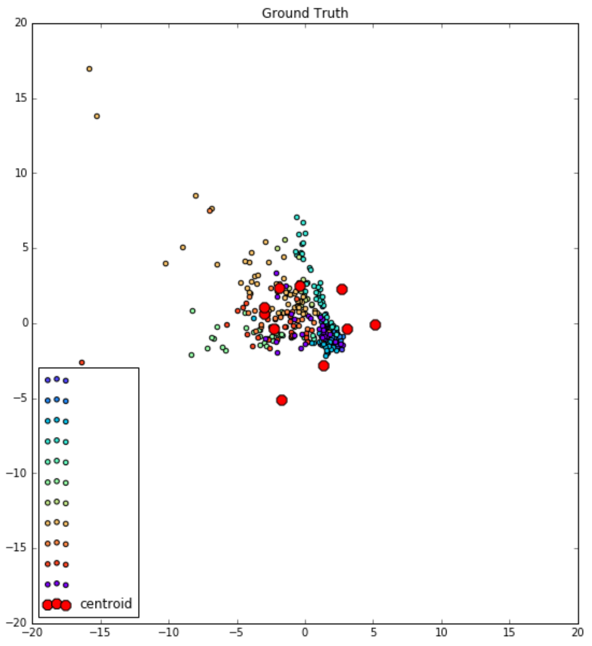
The following is a visualization obtained after running the k-means clustering algorithm on the datasets provided to us:

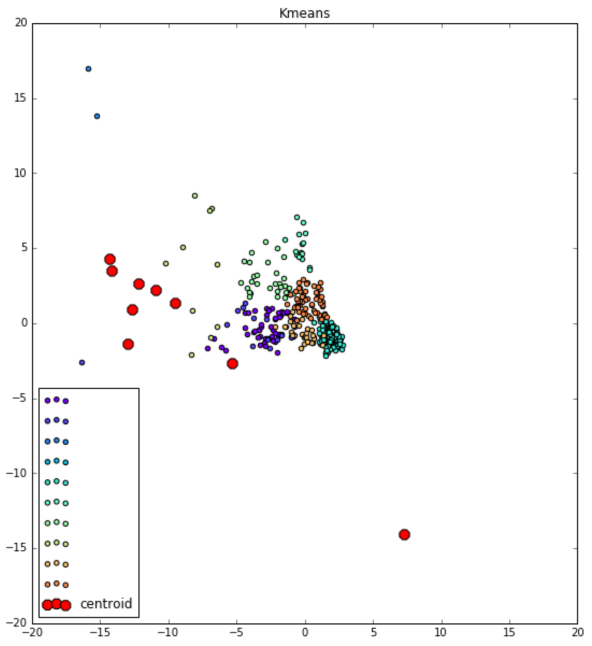
For the dataset Cho.txt





For the dataset Iyer.txt:





Result Evaluation:

External Index/ Datasets	**Cho.txt**	**Iyer.txt**
Rand	0.807793497812	0.784173684663
Jaccard	0.427743585645	0.295929700372

Hierarchical Clustering:

Hierarchical clustering, also known as Hierarchical Cluster Analysis or HCA in the field of data mining and statistics is a method of cluster analysis which aims at building a hierarchy of clusters. There are two approaches for performing hierarchical clustering:

- Agglomerative.
- Divisive.

Agglomerative: This is a "bottom up" approach wherein every object is a cluster in itself. The iteration part of the cluster involves merging two clusters which are most similar to each other and keep repeating this step until all objects are merged to obtain a single cluster.

An example depicting Hierarchical Agglomerative clustering:

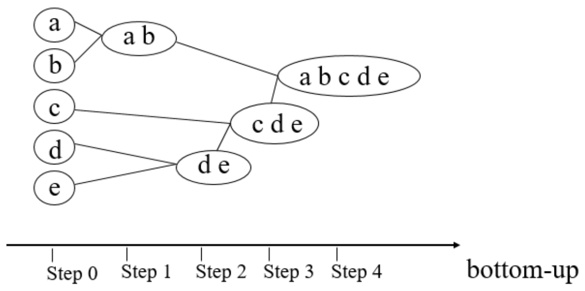


Fig.2 Hierarchical Agglomerative clustering

Divisive: This is a "top down" approach which starts with the all objects clubbed in a single cluster. As one moves down the hierarchy we select a cluster and split it into two sub clusters until each leaf cluster contains only one object.

An example depicting Hierarchical Divisive Clustering:

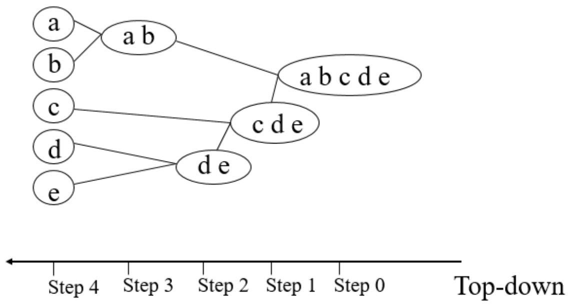


Fig.3: Hierarchical Divisive Clustering

Dendrogram:

The results of hierarchical clustering are usually presented in a dendrogram, which is nothing but a tree that shows how clusters are merged/split hierarchically. In a dendogram, every node of the tree represents a cluster and each leaf node is a singleton cluster.

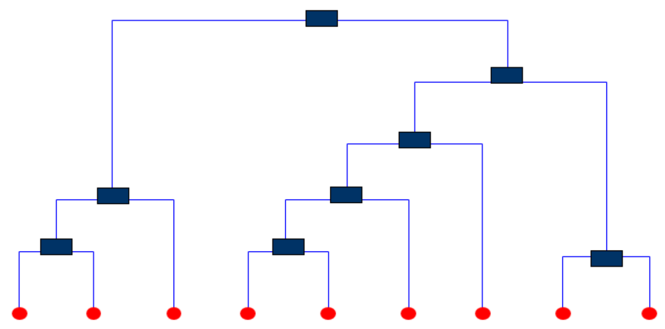


Fig.5 A Dendrogram representation

Clustering of the data objects in a Dendrogram can be obtained by slicing the dendrogram at desired levels where each connected component then forms a cluster.

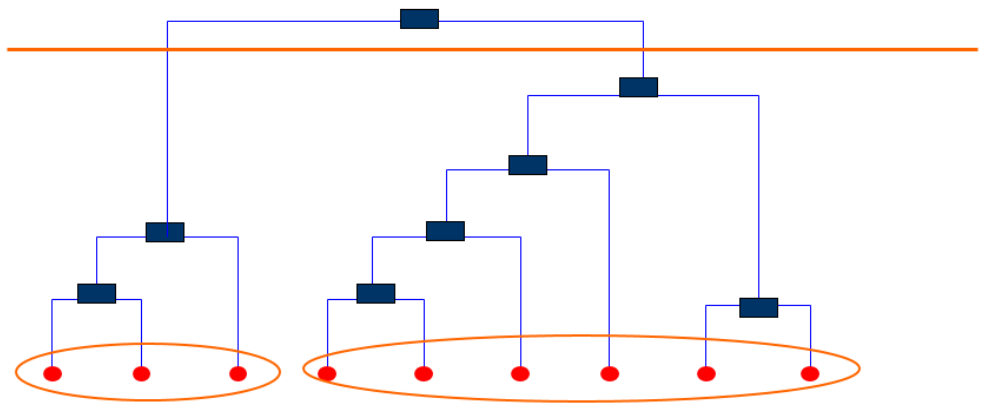


Fig.6 Clustering of the data objects in a dendrogram

Advantages:

- The greatest strength of the hierarchical clustering is the flexibility of obtaining a desired number of clusters by slicing the dendrogram at a proper level.
- They may correspond to meaningful taxonomies- shopping websites- electronics (computer, camera), furniture, groceries.

Disadvantages:

- Once two clusters are combined, it is difficult to undo the decision.

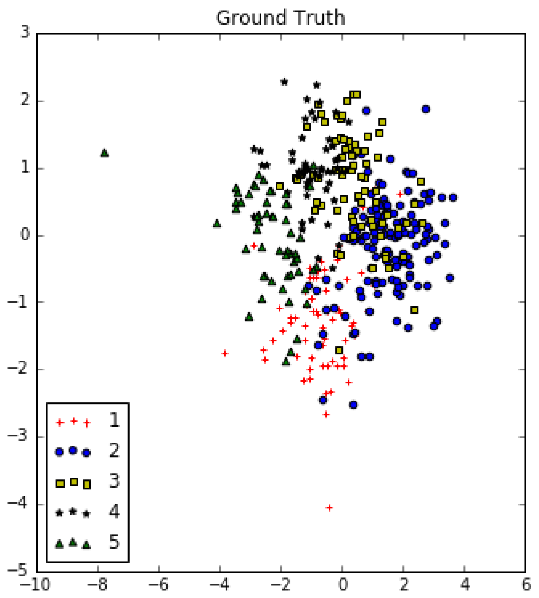
- Directly minimizing the objective function is not possible
- Different schemas are prone to different problems such as:
- Sensitivity to noise and outliers
- Difficulty in handling clusters of different shapes and sizes
- Breaking large clusters into smaller ones

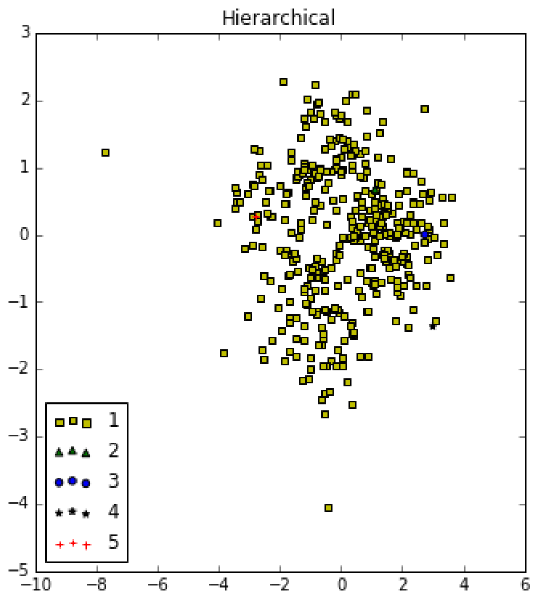
Our Implementation:

1. Coding Language: Python
2. Read the file and split the data into gene data, labels and data_id.
3. Accept number of clusters from the user
4. Function calScore that calculates RAND and JACCARD scores.
5. The main hierachical logic:
   a. Every point is initially labelled as a cluster
   b. Distance of every cluster(point) was calculated against every cluster(point)
   c. Two nearest clusters are merged to form a single cluster.
   d. The combined cluster's distance is calculated as the minimum of the 2 old clusters and the old distance of the individual clusters against every other cluster is deleted.
   e. Repeat steps c-d till only a single cluster is left.
   f. Cut at nth level if you want n clusters.
6. Finally, we performed dimensionality reduction using PCA and displayed the results.

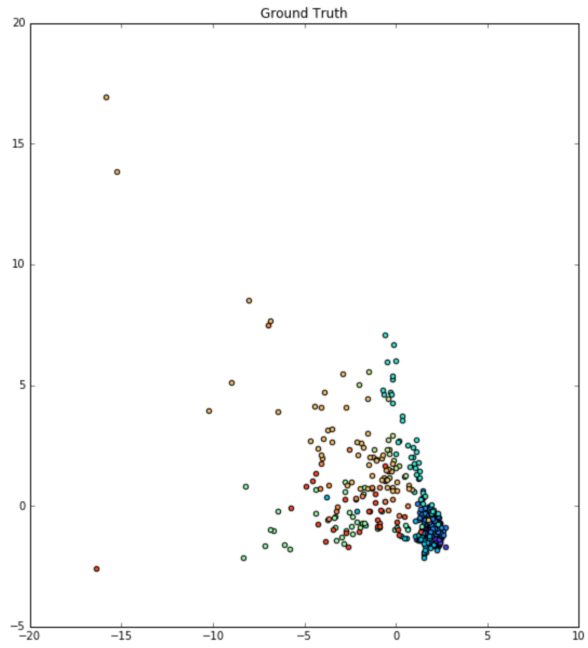
Result Visualization:

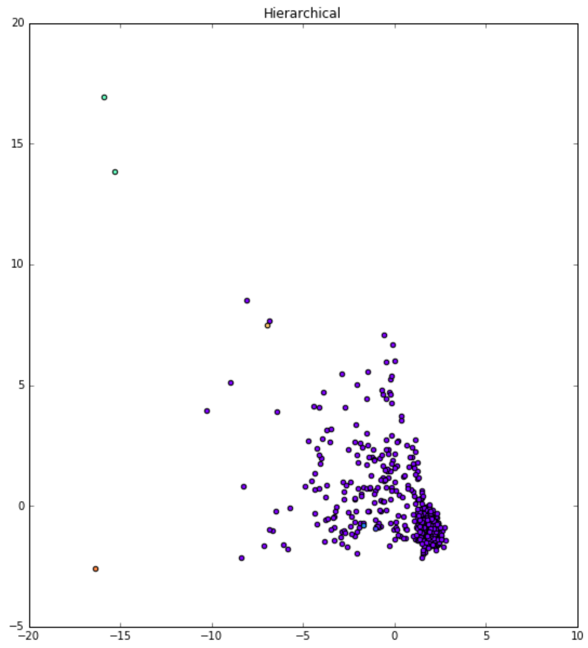
For the dataset Cho.txt:





For the dataset iyer.txt:





Result Evaluation:

External Index/ Datasets	**Cho.txt**	**Iyer.txt**
Rand	0.240274906709	0.188286835597
Jaccard	0.228394977574	0.158243096966

Spectral Clustering:

The methods of clustering discussed so far had limitations associated with them such as:

- The efficiency of k-means is very poor as it can only find spherical clusters and
- Density based approaches are sensitive to parameters discussed above.

Thus, in order to cluster complex shapes efficiently we apply the Spectral approach that uses the similarity graphs to encode local neighborhood information where data points are the vertices of the graph and then we connect the points which are close.

Thus, in multivariate statistics and the clustering of data, spectral clustering techniques make use of the spectrum (eigenvalues) of the similarity matrix of the data to perform dimensionality reduction before clustering in fewer dimensions.

Similarity Graph:

Similarity graph is nothing but a weighted graph representation of the dataset where all vertices can be reached from each other by a path form a connected component. A similarity graph containing only one connected component is said to be fully connected.

$V=\{x_i\}$ Set of *n* vertices representing data points

$E=\{W_{ij}\}$ Set of weighted edges indicating pair-wise similarity between points

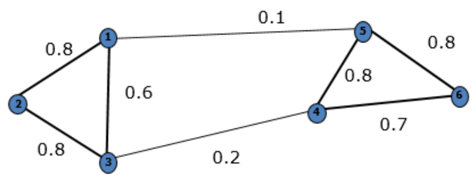


Fig7. A similarity graph

Spectral Clustering Algorithm

The algorithm can be broadly classified into three main steps:

- Pre-processing
  In this process we create a Laplacian matrix L.
  L = D – W Where
  W = Similarity Matrix
  D = Degree Matrix
- Decomposition
  We perform Eigen Vector decomposition to reduce the dimensions.

- Clustering
  We can call any clustering algorithm to cluster this reduced data.

Advantages:

- Spectral clustering works on the eigenvalues hence is more suited for image processing.
- Spectral clustering is more concerned about the connectivity than geometrical proximity so if the data is not that well geometrically separated and clusters aren't connected, the clustering will work fine.
- Spectral clustering is semi-convex while k-means is not.

Disadvantages:

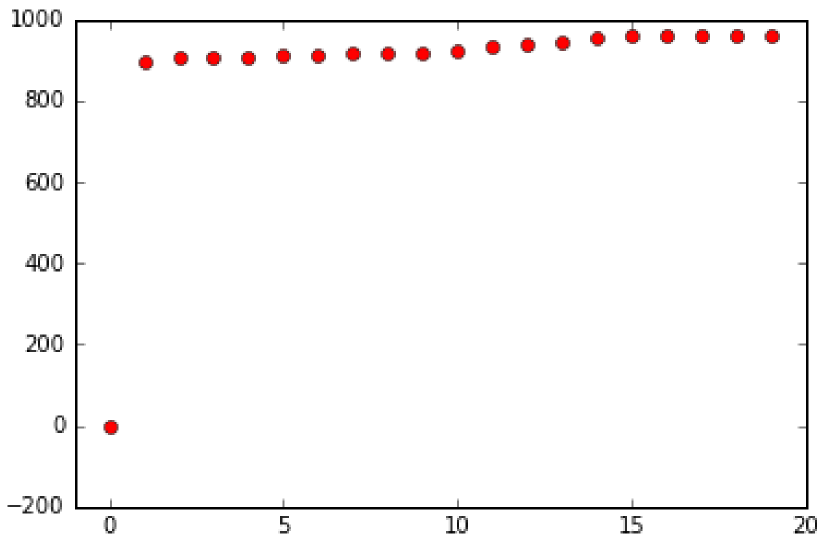
- Based only on local information, the normalized cut functional is not a suitable measure for the quality of clustering.
- Even with a suitable similarity measure, the first few eigenvectors of such adjacency matrices cannot successfully cluster datasets that contain structures at different scales of size and density.

Our Implementation:

1. Coding Language: Python
2. Read the file and split the data into gene data, labels and data_id.
3. Accept number of clusters from the user
4. Function calScore that calculates RAND and JACCARD scores.
5. The main spectral logic:
   a. Distance of every point was calculated against every point. This is the W Matrix.
   b. Next the D Matrix is created. D is basically a diagonal matrix with its diagonal entry equal to the sum of the corresponding row in W Matrix.
   c. L or Laplacian Matrix = D - W
   d. We then did Eigen Vector Decomposition and sorted the eigen values in ascending order.
   e. Difference of every eigen value with its immediately higher value was calculated and the points with the maximum difference was identified as n and n-1.
   f. We then chose n eigen vectors corresponding to the n smallest eigen values and this was our reduced data set.
   g. We then called our own k-means code to perform clustering on this reduced data set.
6. Finally, we performed dimensionality reduction using PCA and displayed the results.

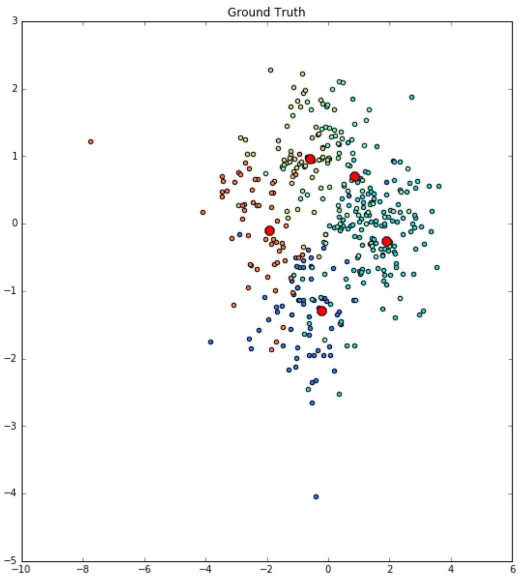
Result Visualization:

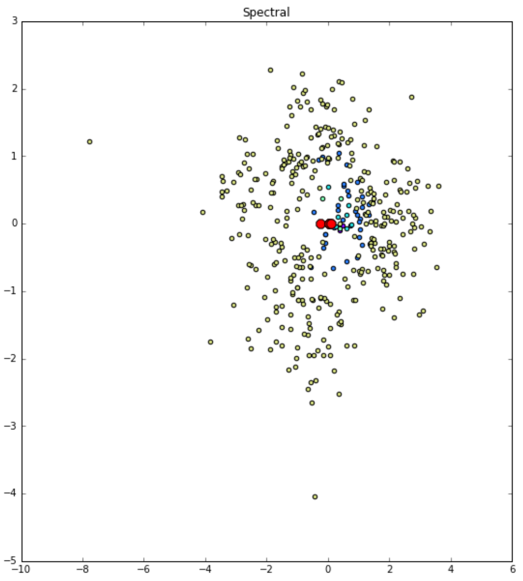
Below is the eigen value plot that was used to perform the dimension reduction process:



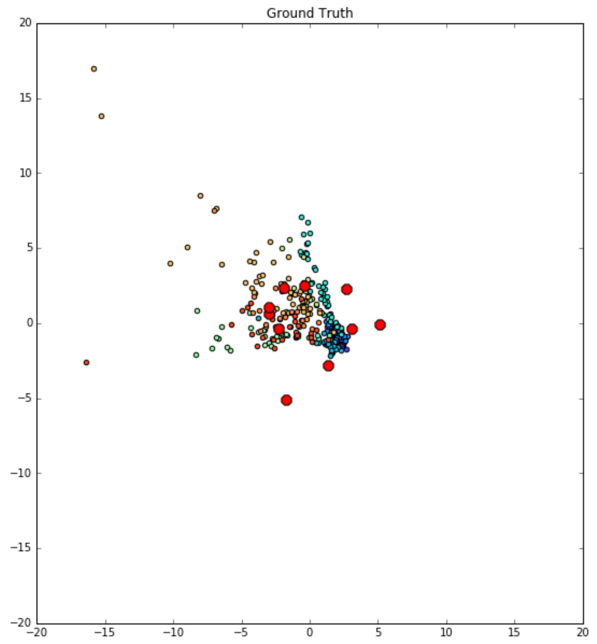
As can be seen in the image, the largest neigbour difference is at at n = 2 & n = 1 and thus we used the first 2 eigen vectors. This entire process is automatic and the visualization is just there for the ease of understanding the process.

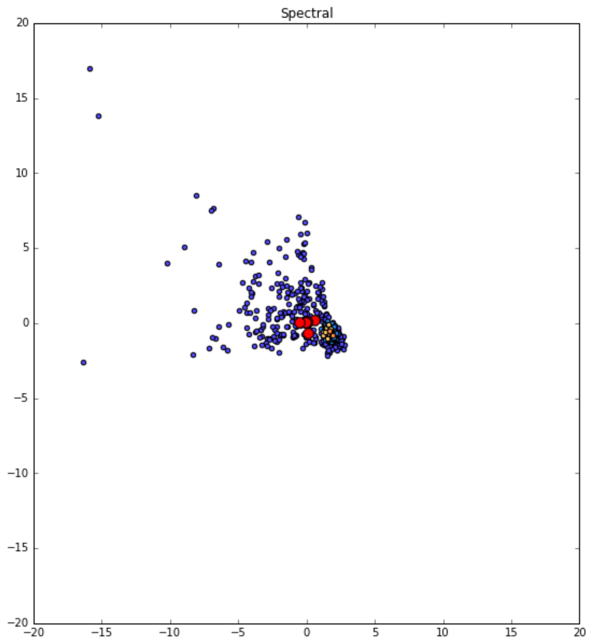
For the cho.txt dataset:





For the iyer.txt dataset:





Result Evaluation:

External Index/ Datasets	**Cho.txt**	**Iyer.txt**
Rand	0.519611264732	0.529610272028
Jaccard	0.194743829174	0.119309625043

Comparison of K-means vs. Hierarchical Agglomerative vs. Spectral

External Index/ Clustering Methods	**Dataset**	**K-means**	**Hierarchical Agglomerative**	**Spectral**
Rand	**Cho.txt**	0.807793497812	0.240274906709	0.519611264732
	Iyer.txt	0.784173684663	0.188286835597	0.529610272028

The above table shows an overall comparison of different clustering techniques on ‘Cho.txt’ and ‘Iyer.txt’ dataset taking into consideration the external index ‘Rand’. From the above comparison it can be seen that ‘K-means’ give a ‘Rand’ value of 0.807 & 0.784 for ‘Cho.txt’ and ‘Iyer.txt’ respectively which is higher than any other clustering algorithm used.

MapReduce K means:

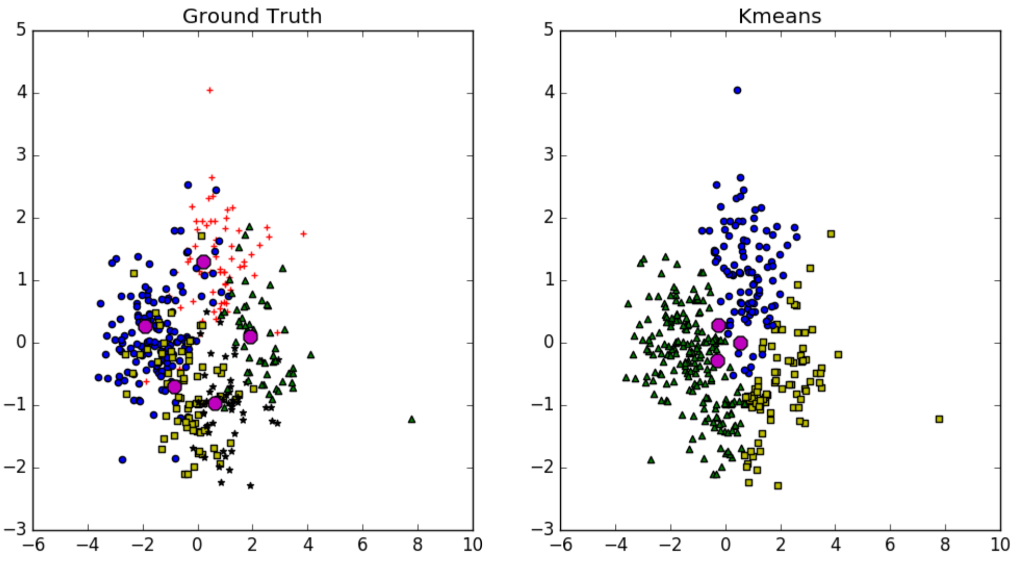
Our Implementation:

1. Coding Language: Python, Unix and Hadoop Streaming API
2. Our code is divided into 5 files:
    a. start.sh
    This shell script performs cleaning operation, ie, it removes the temporary files that we generated in the precious run.
    b. file_process.sh
    This shell script copies the multiple files generated by the reducers. It also performs some cleaning operations. Finally it combines the files generated by the reducers for the next iteration of the mapper.
    c. kmeans.py
    This python file is our wrapper script or the entry point. We have performed input file reading, random centroid assignment, calling the map reduce code using Hadoop streaming api, calling the scripts and finally displaying the ouput.
    d. mapper.py
    Mapper calculates the data points distance against every centroid and assigns the data point to a cluster.
    e. reducer.py
    Reducer recomputes the centroid and ouputs the new centroids and the cluster assignment. If you run n reducers, n files are generated as per MapReduce standards. The task of combining these files falls to our shell script file_process.sh.
3. Finally, we performed dimensionality reduction using PCA and displayed the results.

Comparison with non-parallel kmeans:

1. We have written on entire code in python.
2. We have use numpy module to process the data and it is very efficient as it performs in-memory calculations.
3. When we run kmeans using map-reduce, our python code calls Hadoop and this is a cross server call which is expensive when compared to the small amount of time required to process the small dataset that we are using.
4. Thus due to efficiency of Numpy and the server latency, our non-parallel kmeans gives results faster than our MapReduce kmeans. However, if we were to use the codes on a larger dataset, MapReduce would definitely outperform non-parallel kmeans.
5. We have designed our Mapper and Reducer such that you can call 1-n instances of Mappers and Reducers and it will work smoothly.

Result Visualization:



Result Evaluation:

External Index/ Datasets	Cho.txt
Rand	0.714462133212
Jaccard	0.368577280418

Conclusion:

Even though kmeans gave the best results out of the three, there are many other algorithms as well. Also, for hierarchical clustering, had we have used the Maximum distance as the similarity measure, the results would have been higher than that of k-means. Thus, for any new dataset, you cannot be sure which method would give better results and thus you can sample the dataset and try all the clustering methods. You can then choose the best method depending on the complexity and accuracy that you want.

References:

- Nadler, Boaz, and Meirav Galun. "Fundamental limitations of spectral clustering." Advances in neural information processing systems. 2006.
- https://en.wikipedia.org/wiki/K-means_clustering
- https://datasciencelab.wordpress.com/2013/12/12/clustering-with-k-means-in-python/
- http://nlp.stanford.edu/IR-book/html/htmledition/hierarchical-agglomerative-clustering-1.html
- https://en.wikipedia.org/wiki/Hierarchical_clustering
- https://en.wikipedia.org/wiki/Spectral_clustering
- https://charlesmartin14.wordpress.com/2012/10/09/spectral-clustering/
- http://stanford.edu/~cpiech/cs221/handouts/kmeans.html